1983-06

The database management module of the SPLICE system

Dixon, Ella Jean

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

THE DATABASE MANAGEMENT MODULE OF THE SPLICE SYSTEM

by

E. Jean Dixon

June 1983

Thesis Advisor: Norman F. Schneidewind

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) The Database Management Module of the SPLICE SYSTEM		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis June, 1983
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) E. Jean Dixon		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June, 1983
		13. NUMBER OF PAGES 58
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) SPLICE SYSTEM, Database		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SPLICE (Stock Point Logistics Integrated Communications Environment) is a plan designed to automate data handled at Stock Points and Inventory Control Points for the United States Navy Supply System. The SPLICE concept involves the use of a number of Local Area Networks which communicate via the Defense Data Network. As a part of the ongoing research in the implementation of SPLICE, this (Continued)

ABSTRACT (Continued) Block # 20

thesis addresses the Database Management Module of the Local Area Network and possible problem areas which may be encountered when this module is finally in place. A proposed conceptual design of the database is presented and database computers are evaluated for possible use in SPLICE.

The Database Management Module of the SPLICE SYSTEM

by

E. Jean Dixon
Lieutenant, United States Navy
B.S., Lander College, 1974

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1983

ABSTRACT

SPLICE (Stock Point Logistics Integrated Communications Environment) is a plan designed to automate data handled at Stock Points and Inventory Control Points for the United States Navy Supply System. The SPLICE concept involves the use of a number of Local Area Networks which communicate via the Defense Data Network. As a part of the ongoing research in the implementation of SPLICE, this thesis addresses the Database Management Module of the Local Area Network and possible problem areas which may be encountered when this module is finally in place. A proposed conceptual design of the database is presented and database computers are evaluated for possible use in SPLICE.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. AUTOMATION OF A SUPPLY SYSTEM

Out of a need for a data automation system which would handle stock points and inventory control points, grew a need for a plan to bring together a number of information systems centered around stock point and inventory control point applications. The Stock Point Logistics Integrated Communications Environment (SPLICE) concept is that plan. This concept involves the distribution of a number of local area networks (LANs) which communicate via the Defense Data Network (DDN). This thesis will take a brief look at some aspects of the plan. But before examining the SPLICE concept, we will compare Local Area Networks and Long Distance Networks.

B. LOCAL NETWORKS

A local area network is a data communications system which allows a number of independent devices to communicate with each other, including computers,terminals, mass storage devices, printers, plotters and copying machines. A local network supports a wide variety of applicatins such as file editing and transfer, graphics, word processing, electronic mail, database management and digital voice [Ref. 1]. Each LAN in SPLICE will be uniquely configured and may include some or all of the above components. The question to ask concerning local area networks is "What are the characteristics which make up a local area network?"

According to A. S. Tanenbaum, reference 2, local networks have three distinct characteristics:

1. A diameter of not more than a few kilometers
2. A total data rate exceeding 1 Mbps
3. Ownership by a single organization

Long distance networks, on the other hand, are networks like DDN. A long distance network is usually owned by a communications carrier and is operated as a public utility for its subscribers, providing services such as voice, data and video [Ref. 1]. The way a communications system will allow effective message exchange between different communities of users within each local area network is beyond the scope of this thesis.

C. SPLICE AND ITS RELATIONSHIP WITH UADPS-SP

When SPLICE is examined, we see that it is designed to augment the existing Navy stock point and inventory control point ADP facilities which support the Uniform Automated Data Processing System-Stock Points (UADPS-SP) [Ref. 3]. This system was one of the first attempts at standardizing distributed logistical information. The evolution of UADPS-SP will be traced in the following sections.

1. Origin of UADPS-SP

The original concept of the distributed processing of supply transactions, along with the maintenance of stock records, was first tested at NSC Norfolk in 1956. Upon the successful completion of tests, a number of computers of various sizes and models were installed at a few NSC's (Oakland, Bayonne, San Diego), at NSD Newport and at NSY Charleston in 1957 and 1958. Prompted by a push for standardization of DOD logistics management systems, in February of 1961, the Bureau of Supplies and Accounts (presently NAVSUP) established a full-time committee to standardize procedures as well as equipment at major stock points. The

IBM 1410 computer was selected for Stock Point UADPS in January of 1962. Upon completion of an ADP programming training course, each participating stock point was assigned the task of analysing and programming a particular application. Figure 1.1 lists the initial applications along with

HISTORY OF STOCK POINT UADPS

Application	Title	Activity
A	Requisition History and Status	NSC Bayonne
B	Receipts/Dues	NSD Newport
C	Demand Processing	NSC Norfolk
D	Inventory Control File Maintenance	NSC Oakland
E	Financial Inventory Control	NSCs San Diego and Oakland
F	Stores Accounting	NSC Pearl Harbor
G	Cost Accounting	NSC San Diego
K	Payroll	NSC Pearl Harbor (later changed to NSY Long Beach)

Figure 1.1 Initial Applications.

the activity they were assigned to in 1962. To this list, a number of other applications have been added to date. Of course, this was only the beginning of a system which has grown and evolved over the years. There have been numerous

modifications and alterations to the hardware and software of this system. There has also been a number of subsystems added to it. A list of baseline UADPS-SP application segments or significant subsystems have been implemented by the activities seen in figure 1.2

Today, the hardware for the UADPS-SP consists of the Burroughs medium sized (B-3500/3700/4700/4800) systems. Presently, there are twenty new application systems being develcped which require considerable interactive and telecommunication support. The current UADPS-SP cannot support these requirements without a total redesign effort and will probably require future replacement of the current mainframes [Ref. 3]. Nevertheless, as the Navy Supply System evolves to meet the changing fleet needs, FMSO will adapt and adjust UADPS-SP to meet these emerging requirements [Ref. 4].

2. SPLICE, a computer network in support of UADPS-SP

Returning to the SPLICE concept, it has been decided that the Burroughs computers will provide background processing functions for large file processing and report generation, [Ref. 3]. These are the same computers used in the UADPS-SP system.

According to reference 3,

> SPLICE will be developed, however, using a standard set of minicomputer hardware and software. This standardization is particularly important because SPLICE will be implemented at some sixty different geographical locations, each having a different mix of application and terminal requirements. Additionally, each LAN must have the capability of communicating with other LANs via the Defense Data Network (DDN), which is to be provided by the Defense Communications Agency (DCA).

A layout of the local area network (LAN) can be seen in figure 1.3 . Figure 1.4 , on the other hand, shows the logical network concept. Each local area network will include the following software modules:

1. Local Communications (LC)
2. National Communications (NC)
3. Front-End Processing (FEP)
4. Terminal Management (TM)
5. Data Base Management (DBM)
6. Session Services (SS)
7. Peripheral Management (PM)
8. Resource Allocation (RA)

The above modules will be divided into those modules which perform operating functions and those which support the effective use of these modules on the local area network.

D. STANDARDIZATION OF SPLICE BY DOD

One of the objectives of DOD in SPLICE has been that of standardization. Independent development of local area networks would cause problems. The major problem would be unnecessary duplication of effort and continued production of unique hardware and software. A standard system, on the other hand, would be more economical to design, develop, maintain and operate. For a project the size of SPLICE, standardization is the only wise choice.

E. FUNCTIONS OF THE DATABASE MANAGEMENT MODULE

As a result of ongoing research in the implementation of SPLICE, this thesis will address those issues involved with the design of the Data Base Management Module of the local area networks. The functions performed by the Database Management Module as outlined in reference 3 will be the following:

1. File creation
2. File update
3. Query processing and data retrieval
4. Data dictionary creation and maintenance
5. File catalog creation and maintenance

Figure 1.5 gives an outline of a back-end database management system taken from [Ref. 3]. The implementation of SPLICE for the Database Management Module has been discussed in reference 3 and the conceptual employment of it according to reference 5 is:

> "The concept employed in the recommended implementation of the database and Terminal Management Resource requirements for SPLICE center around a highly decentralized and loosely coupled distributed local area network (LAN)."

The processors for each software module within each LAN will be implemented separately.

F. SCOPE OF THESIS

In this thesis, a proposed conceptual design of the database for SPLICE will be presented. It will also address possible problem areas which may be encountered when this module is finally in place and suggestions on how these problems may be resolved. Finally, some hardware suggestions will be made for the future.

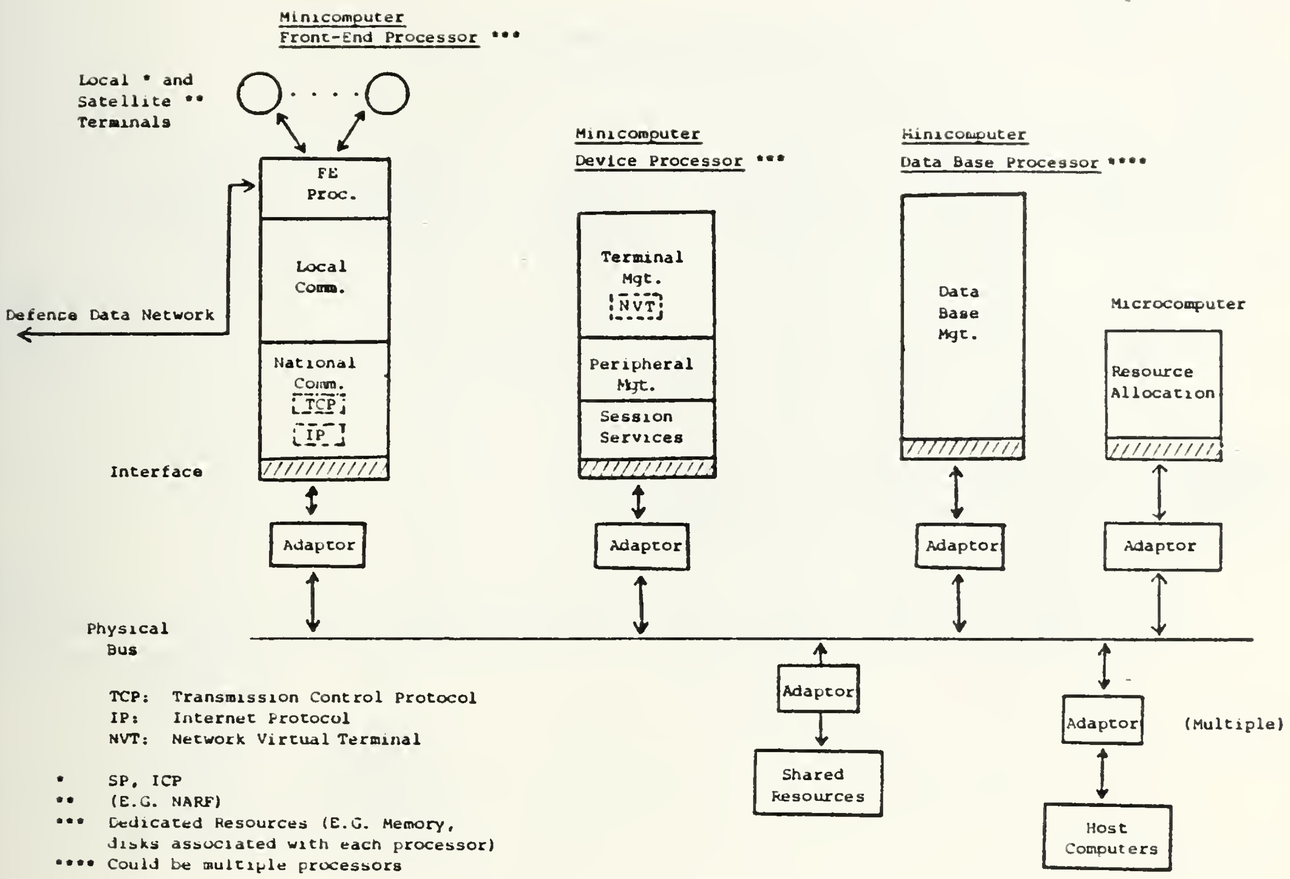
HISTORY OF STOCK POINT UADPS

The following activities have implemented baseline UADPS-SP or significant subsystem/application segments:

UADPS-SP ACTIVITIES

MCAS Cherry Point	NSC Charleston
MCAS El Toro	NSC Long Beach (Disestablished)
MCAS Yuma	NSC Norfolk
NAF Washington	NSC Oakland
NAS Alameda	NSC Pearl Harbor
NAS Atlanta	NSC Puget Sound
NAS Barbers Point	NSC San Diego
NAS Cecil Field	NSD Guam
NAS Corpus Christi	NSD Newport (Disestablished)
NAS Glenview	NSD Subic Bay
NAS Jacksonville	NSY Norfolk
NAS Lemoore	NSY Philadelphia
NAS Memphis	NSY Portsmouth
NAS Miramar	NAVSUBASE Bangor
NAS Moffett Field	NAVSUBASE New London
NAS New Orleans	NAVSUBASE Pearl Harbor
NAS Norfolk	ASO Philadelphia
NAS North Island	NPFC Philadelphia
NAS Patuxent River	NARDAF Newport
NAS Pensacola	NAVMTO Norfolk
NAS Point Mugu	NAVRESUPPOFC New Orleans
NAS South Weymouth	PMOLANT Charleston
NAS Whidbey Island	PMOPAC Bremerton
NAS Willow Grove	SPCC Mechanicsburg
NSC Bayonne (Disestablished)	SWFPAC Silverdale WA

Figure 1.2 Baseline UADPS-SP Application Segments.



TCP: Transmission Control Protocol
IP: Internet Protocol
NVT: Network Virtual Terminal

* SP, ICP
** (E.G. NARF)
*** Dedicated Resources (E.G. Memory, disks associated with each processor)
**** Could be multiple processors

Figure 1.3 Local Area Network Layout.

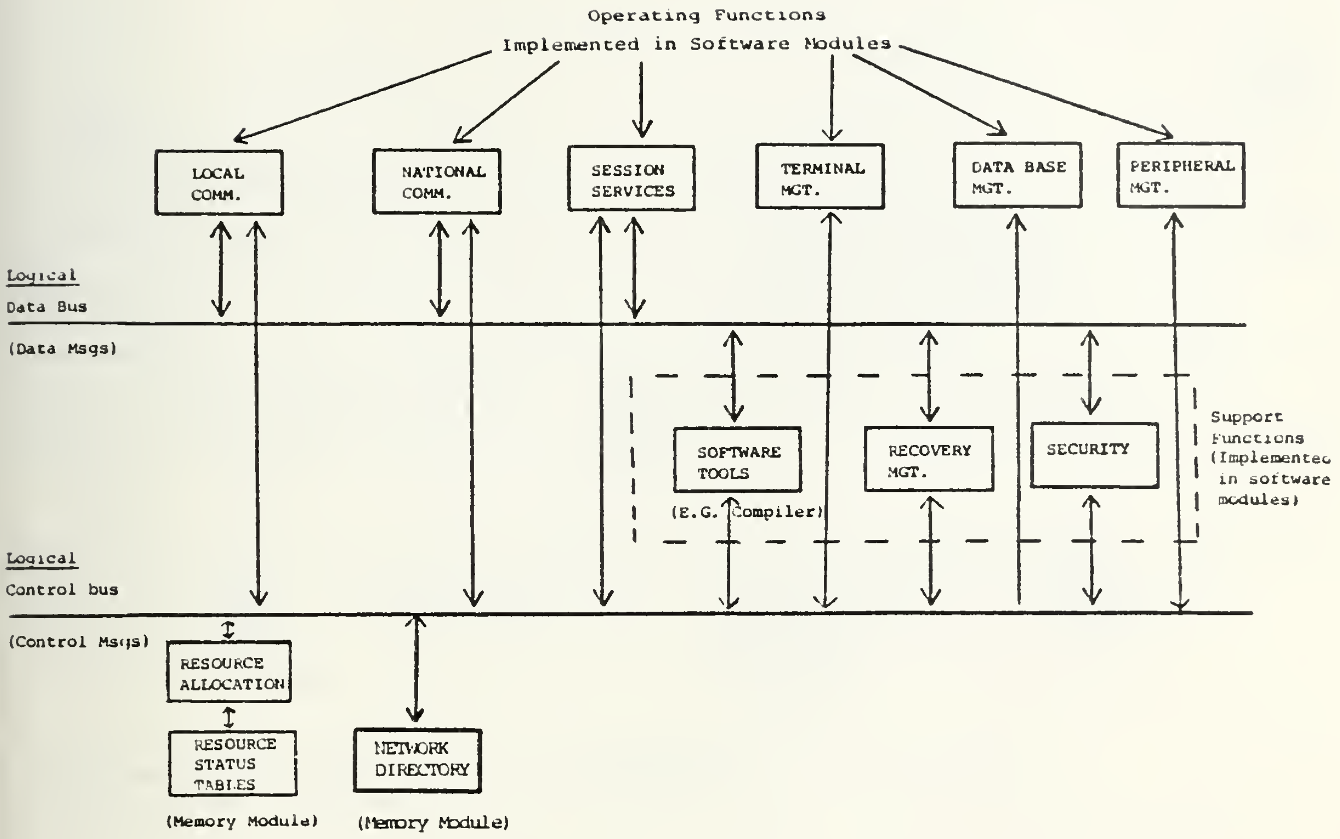


Figure 1.4 Logical Network Concept.

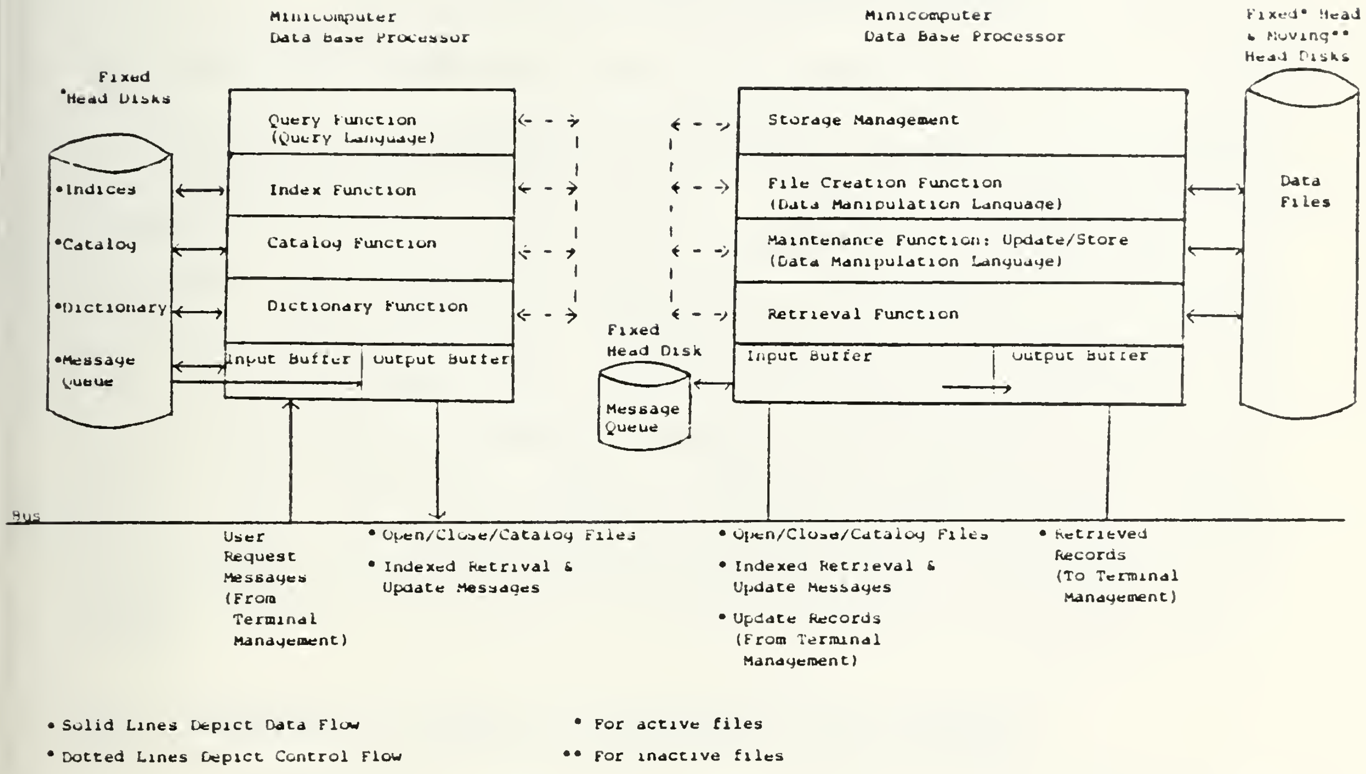

Minicomputer Data Base Processor
Fixed *Head Disks
•Indices
•Catalog
•Dictionary
•Message Queue
Query Function (Query Language)
Index Function
Catalog Function
Dictionary Function
Input Buffer
Output Buffer
Minicomputer Data Base Processor
Fixed* Head & Moving** Head Disks
Storage Management
File Creation Function (Data Manipulation Language)
Maintenance Function: Update/Store (Data Manipulation Language)
Retrieval Function
Input Buffer
Output Buffer
Data Files
Fixed Head Disk
Message Queue
Bus
User Request Messages (From Terminal Management)
• Open/Close/Catalog Files
• Indexed Retrival & Update Messages
• Open/Close/Catalog Files
• Indexed Retrieval & Update Messages
• Update Records (From Terminal Management)
• Retrieved Records (To Terminal Management)
• Solid Lines Depict Data Flow
• Dotted Lines Depict Control Flow
* For active files
** For inactive files

Figure 1.5 Outline of a Back-end Database Management System.

II. THE CONCEPTUAL DATABASE SCHEME

A. DATABASES PRESENTLY A PART OF SPLICE

The present databases of projects under the umbrella of the SPLICE project vary greatly. None of the databases are standard. It is one of the purposes of this thesis to propose a new conceptual design of a database in the SPLICE project which will help standardize database operations for all sites involved with this project. Since each LAN will have a database management module, standardizing databases will allow users to query databases easier from remote sites. All queries could be standardized as well. This chapter will discuss the conceptual design of such a database.

B. DEFINITION OF WHAT A CONCEPTUAL VIEW ENTAILS

The conceptual view is a representation of the entire information content of the database, in a form that is somewhat abstract in comparison with the way in which the data is physically stored...the conceptual view consists of multiple occurences of multiple types of a conceptual record... the conceptual view is defined by means of the conceptual schema, which includes definitions of each of the various types of conceptual record [Ref. 6]. This means that the conceptual view of a database shows the overall content of the database. The conceptual schema defines that view.

1. Definition of SPLICE database

The database with which we are concerned is a database which contains information about parts. These parts are parts for ships, airplanes, etc. Therefore we can assume that basically this database will be a system which inventories parts. In a database of this type certain information is important:

1. Stock number or Manufacturer's number
2. Name of the manufacturer, if it applies to this item
3. The cost of each item
4. The quantity of items available
5. The location of the item, The Activity
6. A brief description of that item.

This is the minimum amount of information which is required for an inventory system.

2. Approaches used to Represent a Database

The next thing to decide is the kind of approach to be used to represent the data in the database. The best known approaches are relational, hierarchical, and network. The approach proposed in this thesis will be the relational approach. The relational approach to data is based on the realization that files that obey certain constraints may be considered as mathematical relations, and hence that elementary theory about relations may be brought to bear on various practical problems of dealing with data in such files [Ref. 7]. Notice the relations given in figure 2.1 These table-like structures are called relations. The rows of such tables or relations are called "tuples" and the columns are usually called "attributes". One concept that relational theory emphasizes and for which there does not seem to be an established data processing term, is the

concept of the domain. A domain is a pool of values from which the actual values appearing in a given column are drawn [Ref. 7].

3. Proposed Conceptual Database Design

In figure 2.1 notice relations of which the database is composed. There are four relations to be considered. The first of these is the Stock-Part relation which includes:

1. Stock Number (Stock-Num) - This number is the Federal Stock number, a thirteen digit number, normally, which is assigned to all stock parts. The stock numbers could be listed in a user's manual which could be placed on secondary storage or online. When the stock number list is updated, an updated version of the user's manual could be printed. (Note: The format of the Federal Stock Number is given below:

   1. Digits 1 - 4 Federal Supply Classification
   2. Digits 5 - 6 National Coding Bureau Number
   3. Digits 7 - 13 National Item ID Number

   Additionally, digits 14 - 15 are used for Weapon Systems and Aviation parts.)
2. Manufacturer Number (Mf-Num) - This is assigned to the part by the manufacturer. Since there is no consistent way of numbering parts by manufacturers, it would be best if we did not use their numbering scheme to inventory parts. There could also be a duplication of manufacturer's

numbers because of the inconsistencies caused by the lack of standards used by manufacturers. The use of Federal Stock numbers would eliminate this problem.

3. Manufacturer's Name (Mf-Name) - The manufacturer's name is given. Some portion of it could be abbreviated.
4. Part Name (Part-Name) - This gives the general category of a part, i.e. rudder.
5. Quantity (Quantity) - This is the number of parts on hand at that particular time.
6. Cost (Cost) - This is the cost of each item.
7. Details (Details) - This will give more details than the part name. Information such as the dimensions of the part (Size, length, etc.) are given. The differences in dimensions will cause the stock number to change, i.e. a 1/2 inch screw has a different stock number than a 1/4 inch screw.
8. Reorder Point (Reord-Pt) - This is the point at which the inventory is replenished for this part. When quantity gets below this point the Vendor's list must be consulted to reorder stock parts (True for Government equipment).
9. Weight (Wt) - This is the attribute which gives the weight of the part in terms of pounds, i.e. pounds/parts.
10. Total Weight (Tot-Wt) - This attribute gives the total weight of all parts with stock number.

Note: The key is Stock-Num.

The second relation is Local-Net. This relation provides fast access to information indicating the sites where stock parts may be found. The attributes included in this relation are:

1. Stock Number (Stock-Num) - Same as stock number attribute in the Stock-Part relation.
2. Database I.D. (DB-Id) - This is the number which will be assigned to each database.
3. Site Number (Site-Num) - This is the number which will be assigned to each site within the SPLICE system.
4. Where (Where) - The location within the SPLICE site of a particular part, e.g. Charleston.

Note: The key is Stock-Num.

The third relation is Vendor-List. It is a list of all vendors who service this LAN. The attributes are as follows:

1. Stock Number (Stock-Num) - Same as stock given in prior relations.
2. Quality Vendor List (QVL) - This is the list of vendors that government agencies are allowed to procure parts from. This list is predefined and could be placed on secondary storage. When a list is needed it could be printed at that time.
3. Bid 1 (Bid1) - Gives the name of the vendor who bid on the parts contract. His bid is also included.
4. Bid 2 (Bid2) - Same as Bid1.
5. Bid 3 (Bid3) - Also the same as Bid1.
6. Bid Evaluation (Bid-Eval) - This attribute lists the vendor who won the bid.
7. Purchase Order (Purch-Ord) - This attribute gives the purchase order number. If this attribute is known, we can collect

historial information or data concerning orders.

8. Lead Time (Lead-Time) - This attribute gives the amount of time which is needed between the time the order is placed and the shipment is delivered.

Note: The key is Purch-Ord.

The final relation is Location-Mf. This relation contains information concerning the manufacturer of the part and includes the following attributes:

1. Manufacturer Number (Mf-Num) - Same as previously described.
2. Manufacturer Location (Mf-Loc) - This attribute gives the city the manufacturer is located in.
3. Address (Address) - This attribute gives the mailing address of the manufacturer.
4. State (State) - This attribute gives the state the manufacturer is located in.
5. Zip (Zip) - This attribute gives the zip code in the manufacturer's address.
6. Phone Number (Phone-Num) - This attribute gives the phone number, which includes the area code, of the manufacturer's representative or saleperson.
7. Salesperson (Salesperson) - This attribute gives the name of the person who sold or was responsible for the sale in a procurement contract.

Note: The key is Mf-Num.

Thus, we have a relational database. Its data are not only informative but also historical in nature. With purchase order numbers, lead time and manufacturer information, it is possible to determine when parts were delivered. The

purchase order number could be placed in the attribute details along with its size, etc. The local network relations could be used to help update parts within the SPLICE system.

Relation Stock-Part has an attribute called weight. This attribute is the weight of a part in pounds/part. When used in conjunction with the total weight attribute, a quick check can be made on the number (quantity) of available parts with this stock number, i.e. Number of parts = (Total Weight)/(Weight in pounds/part). If this number is not the same as the attribute quantity, there is a possible theft or missing part.

A number of queries may be performed on this database. The table-like structures of relational databases make the results of operations performed on a database easy to understand. The operations included in this thesis are:

1. Selections
2. Projections
3. Joins

A selection is an operation which asks for those tuples in a certain relation that meet a certain criterion. For example, using the Vendor-list relation, the Name and Bid of the vendor who made the first Bid on a particular purchase order could be selected.

A projection is an operation which takes a relation, removes some of the attributes, and rearranges some of remaining attributes, if necessary. For example, using the Local-Net relation, if the Database ID and Site Number were needed to answer a query the information could be printed giving the Site Number first followed by the Database ID as opposed to the way it presently appears in the relational table. Therefore data could be formatted in any manner the user wanted.

Finally, there is the join operation. A join is an operation which combines data of two or more relations. For example, using the Stock-Parts and Location-Mf relations, the addresses of manufacturers with a particular stock number could be found using the join operation in conjunction with the selection operation. All of the above operations can be used to answer a query for the user.

The relations in this database are rather long, but they are designed that way in order to avoid having many joins performed on relations. Each relation gives as much information as is necessary for that particular relation. If all of the information is not needed, selections can be made on which attributes should be projected by the user. The user would have no need of joining numerous relations together in order to get the information he needs.

STOCK-PART

Stock-Num	Mf-Num	Mf-Name	Part-Name	Quantity	Cost	Details	Reord-Pt	Wt	Tot-Wt
10...95	17AB	Deer	Bolt	25	00.05	1/8 in.	12	0.05	1.25
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

LOCAL-NET

Stock-Num	DB-ID	Site-Num	Where
45...78	13	2	NAS Charleston
⋮	⋮	⋮	⋮

VENDOR-LIST

Stock-Num	QVL	Bid1	Bid2	Bid3	Bid-Eval	Purch-Ord	Lead-Time
10...95	Deer	Deer	Ace	Sears	Ace	203579	30 days
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

LOCATION-MF

Mf-Num	Mf-Loc	Address	State	Zip	Phone-Num	Sales-person
17AB	Charleston	Apt.	S.C.	29644	803-459-3904	M. Brown
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 2.1 Relational Database Tables.

III. PROBLEMS IN THE LOCAL AREA NETWORK DATABASE MANAGEMENT MODULE

A. POSITIVE CHARACTERISTICS OF A DISTRIBUTED SYSTEM

The local area networks in the SPLICE system are made of distributed systems. Some of the characteristics of a distributed system are given below. First, minicomputers are used in these systems. Secondly, distributed systems give users more individualized control over processing. The scheduling of jobs and quality of services can be determined by the user himself. Finally, distributed systems can be more readily tailored to organizational structures. Since no two organizations are exactly alike, flexibility is important.

B. NEGATIVE CHARACTERISTICS OF A DISTRIBUTED SYSTEM

There are also negative characteristics of distributed systems. First, the procedures required to implement distributed systems are complex. Communications facilities must be procured and computers must be connected. Data compatability also must be ensured. Thus, a number of problems may occur in a distributed environment. These problems and possible solutions will be discussed in the following sections.

C. PROBLEM AREAS

1. Access Control and Security

One of the biggest problems in a distributed system, as in any computer system, is access control and security. If data are online and there are multiple users who may be able to access that data, care must be taken as to how data

are altered. It is very important that the accuracy of data are preserved. Data can be altered in two ways:

1. Accidentally, through typing errors or programming errors
2. Intentionally, through malicious misuse of a database.

To prevent incorrect data from being stored in a database or being read by unauthorized personnel, there are two areas of concern. According to Ullman, these concerns are:

1. Integrity preservation, and
2. Security.

Integrity preservation is guarding against a nonmalicious error. This can be done by writing a program in such a way that it checks for conflicting records before anything is completed (updates, inserts, etc.). Security, or access control as it is sometimes called, is concerned with restricting access of users. Only those persons with a "need to know" should have access to particular data. Therefore, modification and alteration would only be performed by authorized personnel. These precautions, if taken, would allow more control over data and thus preserve the accuracy to a greater extent. As a bear minimum, all online files should have access control using account numbers and accompanying passwords. These passwords would restrict the users to data which is needed only by him or her to get the job done (for example, read-only passwords). This is needed just for control of everyday usage of the database. A number of additional precautions can be taken to further ensure that data are less vulnerable. Encryption, the coding of data so that it is unintelligible, is being used in the SPLICE system as a further precaution for the protection of data. This is not effective, however, unless the programs which encrypt data are protected from would-be infiltrators.

2. Concurrent Updates

A problem associated with all database systems is that of concurrent updates. This is the problem which may occur in any system which has more then one user updating the same file at the same time. In a distributed system, where there are concurrent executions, there is always the chance of there being problems with livelock and deadlock. Livelock is a situation where there may be one or more processes waiting on a locked item. Using a first-come-first-served strategy of locking items usually resolves this problem. Afterwards, all locks are released. Deadlock is a situation in which each member of a set of transactions is waiting to lock an item already locked by another transaction in the set. Since each transaction of the set is waiting, it cannot unlock other transactions. Therefore all of them wait indefinitely. Detection and prevention are two ways of handling deadlocks.

Locks should be placed on all items to be updated before updates are done in order to solve the concurrency update problem. Certain transactions prevent other transactions from accessing a data item until that item is unlocked. Therefore, other users cannot access that portion of the database. This is particularly important in distributed environments because of the various locations of data. If the locking approach is used, the items to be locked should be fairly large, maybe even entire relations. This would reduce some of the costs associated with the locking mechanism.

When viewed over the entire SPLICE system, databases are obviously geographically distributed. For the purposes of maintaining adequate control over cataloging files and maintaining the integrity of related files in the database (synchronization of updating procedures), the database

functions are centralized within each LAN. Other than the fact that some files will remain on the Burroughs Hosts and some files will migrate to an interactive DBM, there is no reason to provide for the distribution of databases within a LAN [Ref. 3].

3. System Crashes

One problem which needs consideration is how to handle or prevent system crashes. When a computer fails, all or part of a transaction may have been completed. There is no sure way to tell exactly what has happened. For that reason, it is essential that backup copies of files be made periodically. For larger databases, copies can be made less frequently than smaller ones because of the amount of time it takes for copying large databases. During recovery after a failure, a determination has to be made as to which transactions should be repeated. For this reason, a log or journal is needed which will contain information concerning all changes to the database since the last backup copy was made. Recovery from failure is particularly a problem with online systems. There may be no copies of the transactions. Therefore, it may be very difficult to recreate the transactions. The reprocessing may not repeat the exact processing sequence.

Finally, consideration should be given to preventing and handling system crashes and of dealing with whether a transaction has been "committed" or not. This is the point at which a transaction is considered complete. When transactions must be "redone" or "undone", a log or journal which contains those which have committed will assist in reconstructing the database. According to Ullman, [Ref. 6] one could define a two-phase commit policy which would operate as follows:

1. A transaction cannot write into the database until it has committed.
2. A transaction cannot commit until it has recorded all its changes to items in the journal.

All unlocking is done after the committed transactions have occurred. Uncommitted transactions cannot be input to the database. If a crash occurs, destroyed data can be redone using the backup copy of committed transactions. A message could be sent to the user warning him about transactions which have not been completed. Also, after a crash, locks may still be in place on data items, a recovery routine will have to remove these locks.

4. Data Location

(1). Global or Local.

The characteristics of data, as related to a distributed system, will now be examined. According to Kroenke, [Ref. 8], the characteristics of data in the distributed environment can best be examined by considering two questions that the designer of a system must answer. First, "Where is the data to be located?" Second, "How will it be updated?"

Data can be either local or global in a distributed system. Local data is only needed at the local node [Ref. 8]. It is processed by an application program of a local computer. Since data is not used by other local computers, nodes never request data from other nodes. Global data, on the other hand, is needed by a program or programs that run on at least two computers in the distributed system [Ref. 8]. According to Kroenke, eighty percent of the data at a node tends to be local and twenty percent is global. These are rough guidelines. Also, Kroenke feels that local data should remain local. Communication costs are too high to move local data from the node on which it is used.

(2). Centralized or Partitioned.

It is not easy to know exactly where to put global data. The first consideration is whether it should be centralized or partitioned. If it is centralized, one computer in the distributed network stores the data. A request is sent to this computer when global data is needed. Partitioned data, however, is spread over several computers. If data is needed , its location has to first be determined and then accessed.

The advantage of centralized data is that every node knows the location of data. This is the approach which will be used in the SPLICE system for the Database Management Module in each LAN. This makes the system simple. The concurrent update problem would only be handled by one processor. However, if global data is needed, it is possible that a performance bottleneck will develop when accessing data from one computer. Partitioned data would not cause this kind of problem.

Reliability also has to be considered. With centralized data, if the one database computer fails, all nodes in the local network would have to discontinue processing. When global data is needed in a partitioned data system if one node fails, all other nodes in the network could continue processing. Unfortunately, updates of partitioned data are much harder to control [Ref. 8]. A centralized system can be configured to continue operation, in the case of failure, by using redundant hardware and software, combined with mirrored disk operation and checkpointing. This will be the way SPLICE will handle this problem.

(3). Replicated or Nonreplicated.

There should be a number of copies of the data stored in the network. To replicate centralized global data, the entire collection of global data is stored at several locations in the network. When this is done, the

nodes need only keep lists of the computers having the replicated data. They do not need directories that show the locations of particular kinds of data; all of the data is located at each of the nodes. Also, replicating global data eliminates the bottleneck and reliability problems discussed above, but introduces the problem of concurrent update control [Ref. 8].

When data is partitioned or replicated, each node must have access to a directory that gives the location or locations of each type of data [Ref. 8]. Therefore, when a user tries to access global data, the operating system or DBMS calls upon the directory to find the location of that data.

Finally, Kroenke states that the greatest amount of flexibility can be found with the replicated, partitioned storage of data across a network. However, control is much more difficult. More complexity is also added to the operation of the network.

When updating data in a system that has replicated data, there is an issue which should be considered. In a centralized, nonreplicated data system, an application program can lock records before they are used. The lock only involves data in a single computer, whereas with replicated data a lock would have to be placed on all computers with the global data item. The problem with this is that if locks are applied simultaneously, one user could possibly have the record locked on one computer while another user has the record locked on another computer. Neither user has complete control because both locks are in place for two different users. The system has to resolve this conflict. This process can waste a lot of time.

Even nonreplicated, partitioned data can cause problems. If a transaction is to be applied to several different records. There is no problem if this update is

done on one computer. If some of the data are on different computers, however, there could be two users locking each other out.

These are concurrency problems. The resolution of these problems is an active research area. Even though the concurrency updating problem is of concern, the scope of this thesis does not include the resolution of such problems.

IV. THE DATABASE MACHINE AS AN ALTERNATIVE DESIGN

A. GROWTH POTENTIAL OF SPLICE PROJECTS

The database computer (DBC) or the database machine (DBM), as it is sometimes called, should be considered as one of the hardware alternatives for implementating the Database Management System (DBMS). By offloading DBMS functions from the host application computers to DBM, application processing speed is increased and SPLICE application growth can be more readily absorbed.

B. CONVENTIONAL COMPUTERS

1. Designed using Large, Complex Software

Large databases of the future may need a DBM. Because the database machine is a special purpose machine, which can handle DBMS efficiently, large databases of the future will more than likely be managed by database machines as opposed to conventional database management software.

In the following paragraphs, DBMs will be examined. Actual models of DBMs will be presented and the different approaches to DBMs architectures will be discussed.

2. Designed to Access Data by Physical Address

Conventional computer architectures and applications have been designed to refer to physical addresses in order to address data. With the increasing number of applications which are centered around information storage and retrieval, the conventional systems are unable to retrieve information by content. This inability to handle content addressing has lead to interest in computer architectures which are more

efficent in informaticn storage and retrieval applications. One cf the solutions to this problem is the database computer. The database computer can be incorporated into a system in one of four ways:

1. Back-end processor for a host
2. Intelligent peripheral control unit
3. Storage hierarchy
4. Network node.

These approaches are independent of each other, which may suggest that more than one approach could be included into a system's architecture.

C. DATABASE COMPUTER APPROACHES

1. Back-end Processor

The back-end processor is a general purpose computer which is thought of as a master-slave configuration. High level access requests are passed to the back-end processor by the host computer. Access validation, management of storage, update lockout, response formatting, and I/O operations are all performed by the back-end processor. After the back-end processor is finished, it passes the response back to the host.

2. Intelligent Peripheral

The intellignet peripheral control unit works in conjunction with a mass storage device. Highly repetitive data accesses are moved to a mass storage controller to avoid high overhead on the host hardware and software. Functions like device scheduling, head positioning, data recovery, searching, sorting and error correction are performed by the intelligent peripheral control unit. Functions like sequential associative access and parallel read (on a disk) can also be implemented. (Note: An

associative computer architecture allows data to be accessed directly by value without physical addresses.)

3. Storage Hierarchy

The storage hierarchy is similar to the cache memory approach in that both are concerned with the locality of data. The locality of access is such that data already used or data near other data which has been used, is very likely to be accessed in the near future [Ref. 9]. This characteristic can be used to speed up the access of data in a database. The database cache could be inserted in the system between main storage and disk [Ref. 9]. If implemented by a sequential access device, such as CCD or bubble storage, access time would be less than 1 msec, if data is located in the cache. Otherwise, the request would take longer. The fact that data is managed on the least-recently-used basis, ensures that most of the active data resides in the fast access CCD or bubble storage.

4. Network Node

According to reference 9, The "network node", yet another approach to a database computer, is a general purpose computer which communicates with several other nodes in the system, most frequently using a data communications protocol and serial channels, but possibly using I/O channels. The benefit in of this configuration is that several nodes (hosts) can access a single shared database, thus avoiding replication of the data. It is implemented on a general purpose system, host and back-end, or processor or host and intelligent control unit.

D. DATABASE TECHNOLOGY

In the past decade, the Database Management System has become more popular. There are many gains to be made through the use of database technology. The Database Management System relieves the application program of many tasks. Yet, the Database Management System has had its drawbacks. Typically, the software laden database management system has been large in size and complex in structure, which not only overtaxes the host hardware, but also stresses the host operating system [Ref. 10]. Large and more sophisticated applications began to demand more speed, capacity and retrieval flexibility on general purpose computers. Therefore, it's not surprising that a way to alleviate some of the demands on general purpose computers has been sought. As technology improved it was clear that a more efficient form of back-end processor could be developed as a "database machine", one specially designed to manipulate and access data with more flexibility than conventional computers running general purpose software [Ref. 11].

E. INITIAL RESEARCH OF DBMS

Research on the DBM concept started at Bell Labs in the early seventies as work on a "back-end" DBMS (Database Management System) using general purpose computers in a dedicated environment [Ref. 11]. As a back-end machine, the DBC (Database Computer) attempts to achieve high performance and low cost [Ref. 10]. Originally, the five goals of the DBC were:

1. To design a machine with the capability of handling a very large online database of 10 bytes or greater (The DBM is usually not cost effective on a smaller database)
2. To build a database computer today (1979)
3. To have the DBC compete in a favorable

manner with the existing DBMS as far as system throughput and cost of database storage were concerned
4. To make security an integral part of the DBC design
5. To provide a repetoire of very high-level commands in order to sufficiently interface with front-end computers and support database management applications.

F. MODELS OF DBMS

1. IDM Family

a. IDM 500/1 and IDM 500/2

Britton Lee, Inc., a company in Los Gatos, California, has developed a number of relational database machines. Among those developed are the IDM 500/1 and IDM 500/2, relational intelligent database machines, as well as the IDM System 300/600, a relational database management system.

The IDM 500/1 was the first high-performance Intelligent Database Machine (IDM) on the market. It serves as an auxillary processor to one or more host computers and is driven by a high-level query language which is resident in the host. It handles the relational database tasks and manages dedicated database disks. The IDM 500/1 has room for expansion for medium to large database applications and is programmed to optimally perform retrieving, updating, sorting, etc. The IDM 500/2 is basically the same machine but it is a high-end custom-designed 10 MIPS database accelerator model. It handles transactions 2 to 10 times faster. A full complement of database management functions performed by both members of the IDM family are listed below:

1. Basic Commands - Host formatted to IDM specifications, which create and destroy databases, relations and indicies to data. Also appending, retrieving and replacing data are handled in relations.
2. Integrated Data Dictionary - Data dictionary relations are automatically maintained on information about data.
3. Transaction Management - Ensures that user-specified transactions are fully completed or backed out in the event of a failure.
4. Concurrency Control - Allows multiple users to safely access the same database simultaneously.
5. Access Control - Protects data by using such features as deny/permit access privileges and read/write locking of shared data.
6. Audit Logging - Maintains check pointed audit or transaction logs for auditing, backup and recovery.
7. Backup and Recovery - Online dump facility supports backup of disks, databases and transaction logs to disk, tape and the host. The database is recovered via a load and transactions are rolled forward.
8. Random Access Files
9. Stored Commands - Minimizes execution time. User-defined stored commands are featured.

All these functions can be seen in 4.1 The architecture is modularized and expandable. In figure 4.2 the data processor, main memory, disk controller and tape controllers (optional) are shown.

Host interfacing is provided by both parallel IEEE-488 and serial RS232C host interface modules (Up to 8 hosts). The system also has extra expansion slots for future growth. The IDM 500/2 has an extra function, the Database Accelerator. It is custom-designed and has an instruction set which optimizes relational database processing.

b. IDM 300/600

The IDM System 300 and IDM System 600 are complete relational database management systems for DEC VAX users running VMS or UNIX, and for PDP-11 UNIX users, 12 Both combine an Intellignet Database Machine (IDM), of Britton Lee, with end-user software tools in the host for database applications. Included in the software are: 1) data entry facilities 2) an ad hoc online query language 3) a report writer 4) program language interfaces for FORTRAN, COBOL and C (programming language) 5) a full complement of Database Administration utilities. The IDM System 300/600 architecture can be seen in figure 4.3

The functions performed by the IDM System 300/600 are the same as those listed above for the IDM 500/1 and IDM 500/2. There is an additional function, however. That function is Multiple Host Support. It can be expanded to allow several hosts to access its databases. This access is provided by the IDM System 300/600 UNIBUS Interface Packages. Figure 4.4 shows the system architecture. Figure 4.5 shows a summary of the maximum IDM capacities.

2. iDBP 86/440

Another consideration is the Database Processor (iDBP 86/440) by Intel. It is a microprocessor-based relational database management system. Functionally, it is a mass storage controller for one or more hosts. Software and specialized hardware are included in the database management

system design. It is positioned between the host(s) and a set of dedicated disks. 4.6 shows the system architecture.

The iDBP provides a database management system "kernel" which supports relational, hierarchical and network databases. It also provides concurrency control, security, integrity and recovery mechanisms for sharing of data. In addition to traditional, record-oriented files, the iDBP also manages unstructured files which may contain text, graphics, digitized voice or digitized images [Ref. 13]. Even though the iDBP 86/440 is the first of a family of database machines by Intel, it will continue to be enhanced as the new VLSI component is integrated into it in the future. Figure 4.7 gives the configurability of the iDBP and its system capacities.

3. NOAH

The final database machine to be examined is called NOAH, produced by HDR Systems Inc. NOAH is a relational database machine which provides a modular architecture. 4.8 gives NOAH's hardware configuration. Some of NOAH's software modules reside on processors dedicated to database management and query language functions while others reside on the general purpose host. The functions performed by NOAH are:

1. Query language (SQL/NOAH)
2. Integrated Data Dictionary
3. Security
4. Recovery

Figure 4.9 gives specifications and configuration information.

These were, of course, only a few of the database machines which have been designed and developed in the last few years.

G. EXPECTED PERFORMANCE OF DBCS

1. Advantages of DBCs

A large number of database management functions are implemented in hardware. Since this is the case, DBC computers are expected to perform quite a bit better than the computers which provide these functions in software. Software security enforcement may also be absorbed in hardware.

An existing database may be supported on the DBC by converting the database to conform to the DBC representation of data. This one time conversion is known as database transformation [Ref. 10]. The DBC manufacturers claim that no reprogramming of the database management application is necessary, unless the user wishes to reformat his data. An interface will translate the database management calls into DBC commands. The interface requires a small amount of software because DBC commands resemble high-level data languages. This process is known as query translation. The DBC interface package resides in the front-end computer. The interface, together with the database computer, replaces a full-scale software database management system and its conventional disk storage [Ref. 10]. The application program, however, is not replaced.

According to reference 10,

> It is estimated that in supporting these applications on the DBC, the database storage requirement is as much as 1.5 or 2 times that in a conventional system. This excess storage requirement, however, is adequately offset by one or more orders of magitude improvement in the execution time of user transactions. Furthermore, the storage requirement for the indicies decreases by one or more orders of magnitude. Finally, the size of the software (i.e. the DBC interface) is expected to be several orders of magnitude smaller than conventional database management software.

Today, software for mainframe computers is handling problems such as recovery from failure, concurrency control, and integrity validation. If the DBC handles such problems, it would relieve the mainframe system of many of the database software functions.

2. Disadvantages of DBCs

In spite of all the potential benefits provided by database computers, there are some disadvantages which must be pointed out. These disadvantages are the following:

1. DBMs increased system complexity
2. DBMs load balancing is difficult
3. DBMs will create training and conversion requirements for users

When a system is designed, complexity, functionality and cost must be considered. A decision has to be made as to which of these issues is most important to an organization. The back-end processor and its associated software adds cost and complexity to the total computer system [Ref. 14]. The decision which must be made by the organization, however, is whether these two added dimensions will pay for themselves in the future. Also to be considered is the fact that two hardware and software systems will have to be maintained.

Another consideration is the fact that with conventional hosts, it is possible to balance the load between computers. Once a DBM has been acquired this is not possible, since the DBM is dedicated to one task, database management. However, it must be mentioned that one of the main reasons for acquiring a DBC is to offload DBMS from the host to the DBC.

Finally, if an organization has never used a database computer before, there will be a considerable amount of time which will be needed for training and system conversion. The organization must consider this and incorporate

its effect into the time it will take for the system to become operational. Also, the organization must anticipate resistence to the new technology from experienced people. These are a few of the possible disadvantages an organization could encounter as a result of changing its operation to include the database computer concept.

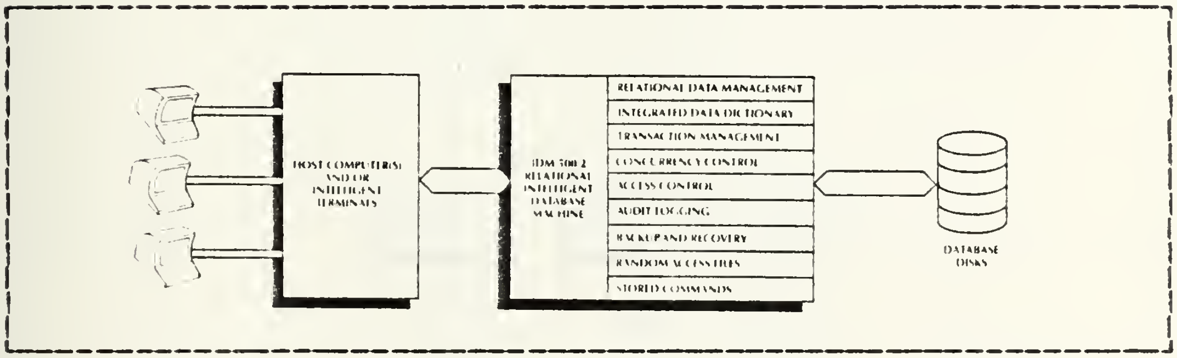

HOST COMPUTER(S) AND OR INTELLIGENT TERMINALS
IDM 500/2 RELATIONAL INTELLIGENT DATABASE MACHINE
RELATIONAL DATA MANAGEMENT
INTEGRATED DATA DICTIONARY
TRANSACTION MANAGEMENT
CONCURRENCY CONTROL
ACCESS CONTROL
AUDIT LOGGING
BACKUP AND RECOVERY
RANDOM ACCESS FILES
STORED COMMANDS
DATABASE DISKS

Figure 4.1 Database Management Functions of the IDM 500.

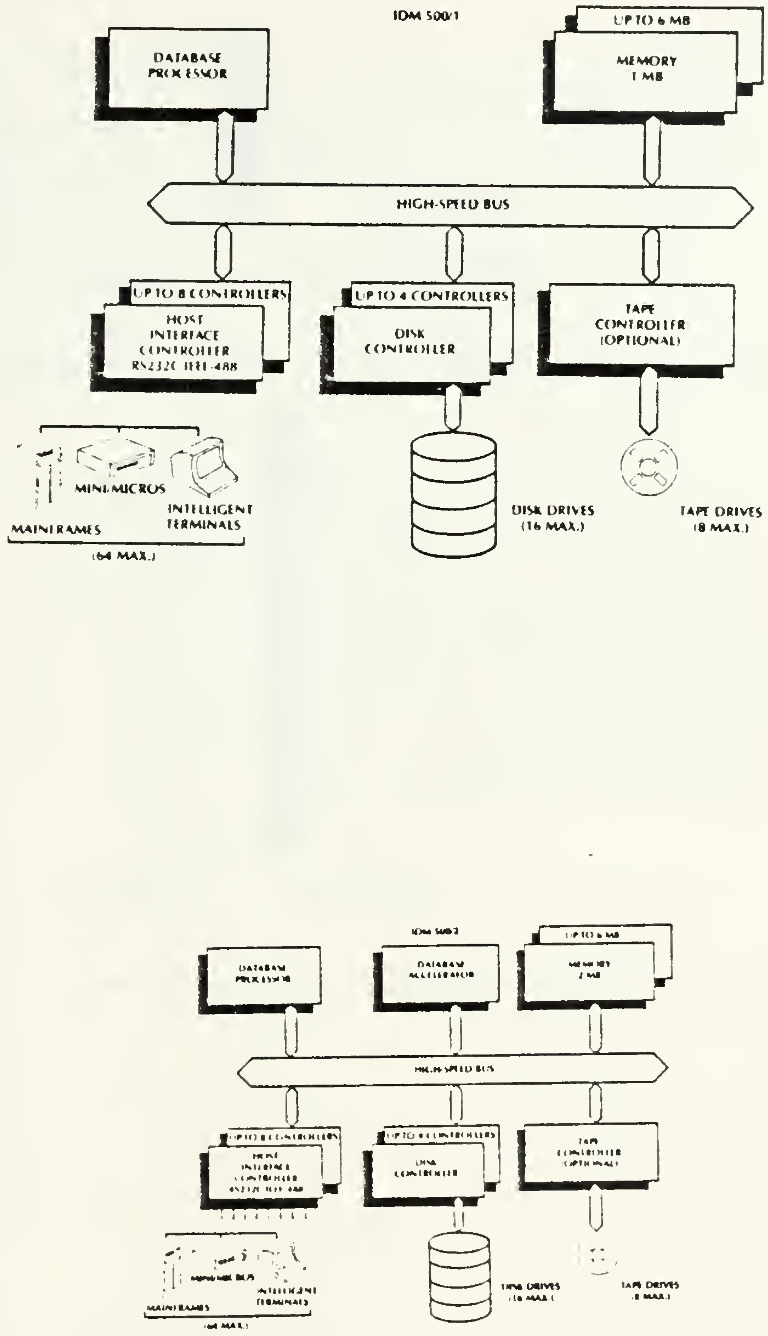


Figure 4.2 Architecture of the IDM 500/1 and 500/2.

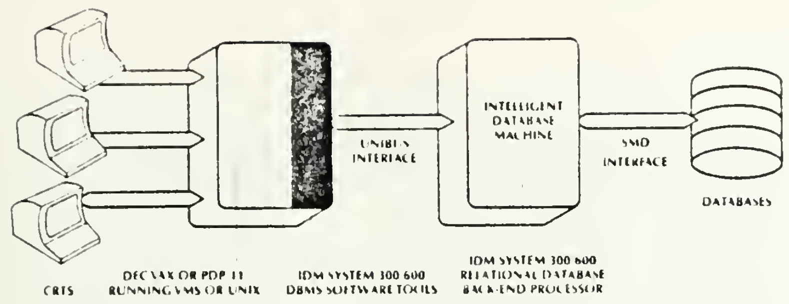

INTELLIGENT DATABASE MACHINE
UNIBUS INTERFACE
SMD INTERFACE
DATABASES
CRTS
DEC VAX OR PDP 11 RUNNING VMS OR UNIX
IDM SYSTEM 300/600 DBMS SOFTWARE TOOLS
IDM SYSTEM 300/600 RELATIONAL DATABASE BACK-END PROCESSOR

Figure 4.3 IDM System 300/600 Architecture.

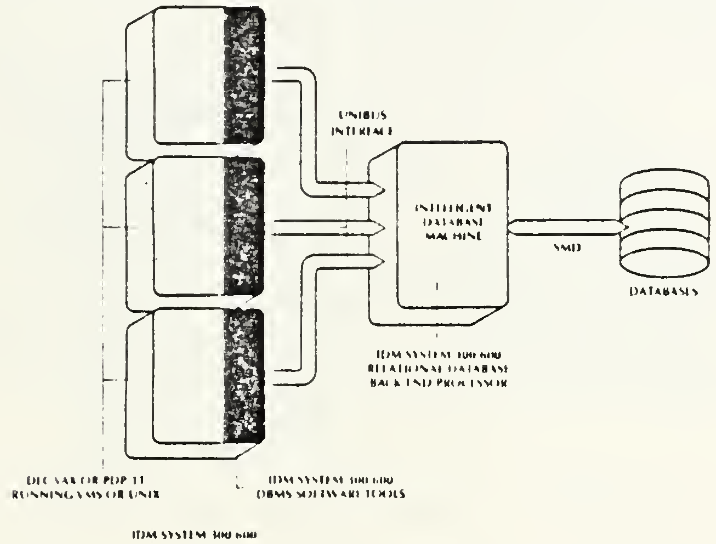
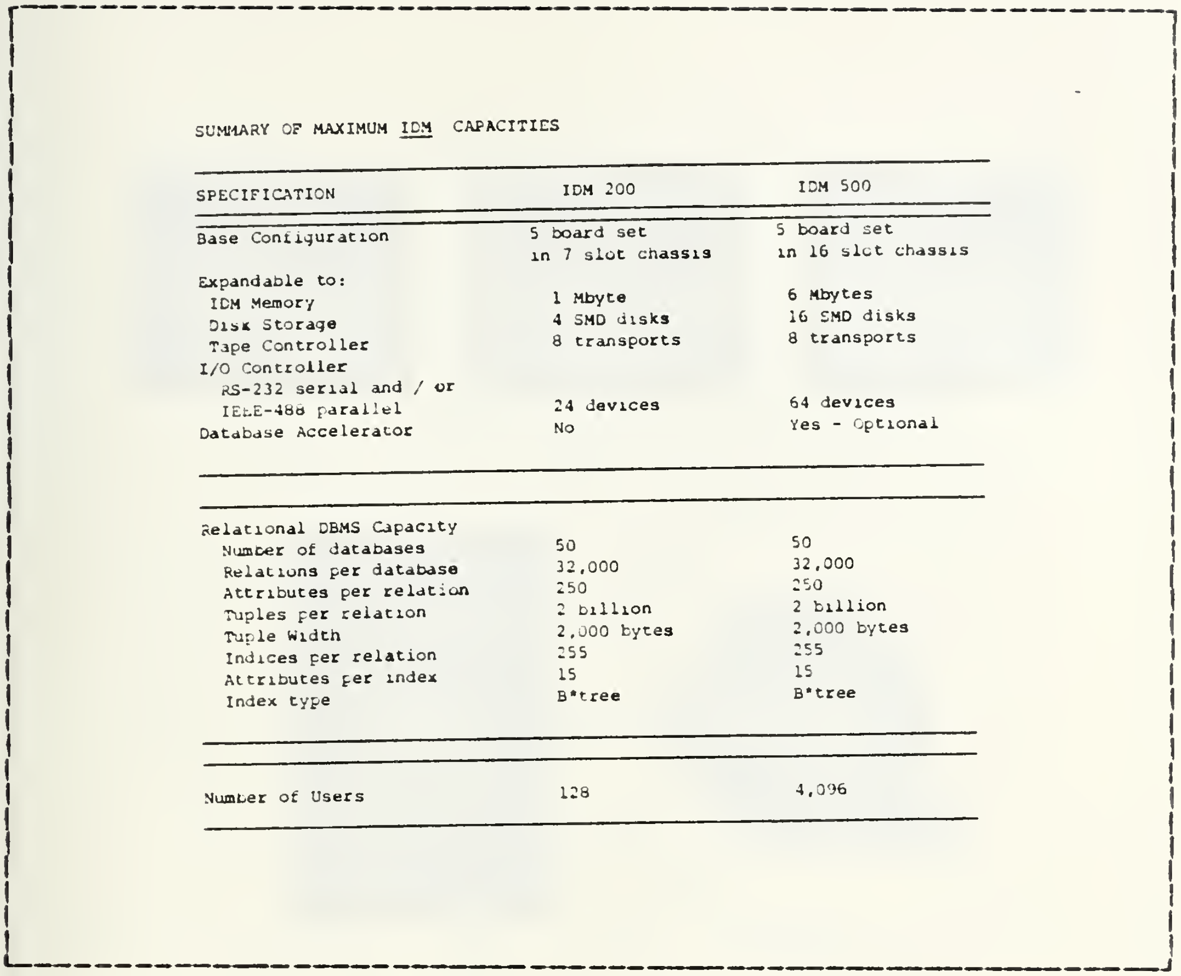

UNIBUS INTERFACE
INTELLIGENT DATABASE MACHINE
SMD
DATABASES
IDM SYSTEM 300/600 RELATIONAL DATABASE BACK END PROCESSOR
DEC VAX OR PDP 11 RUNNING VMS OR UNIX
IDM SYSTEM 300/600 DBMS SOFTWARE TOOLS
IDM SYSTEM 300/600

Figure 4.4 The IDM System 300/600 Architecture with Interfaces.



SUMMARY OF MAXIMUM IDM CAPACITIES

SPECIFICATION	IDM 200	IDM 500
Base Configuration	5 board set in 7 slot chassis	5 board set in 16 slot chassis
Expandable to:		
IDM Memory	1 Mbyte	6 Mbytes
Disk Storage	4 SMD disks	16 SMD disks
Tape Controller	8 transports	8 transports
I/O Controller		
RS-232 serial and / or IEEE-488 parallel	24 devices	64 devices
Database Accelerator	No	Yes - Optional
Relational DBMS Capacity		
Number of databases	50	50
Relations per database	32,000	32,000
Attributes per relation	250	250
Tuples per relation	2 billion	2 billion
Tuple Width	2,000 bytes	2,000 bytes
Indices per relation	255	255
Attributes per index	15	15
Index type	B*tree	B*tree
Number of Users	128	4,096

Figure 4.5 Summary of the Maximum IDM Capacities.

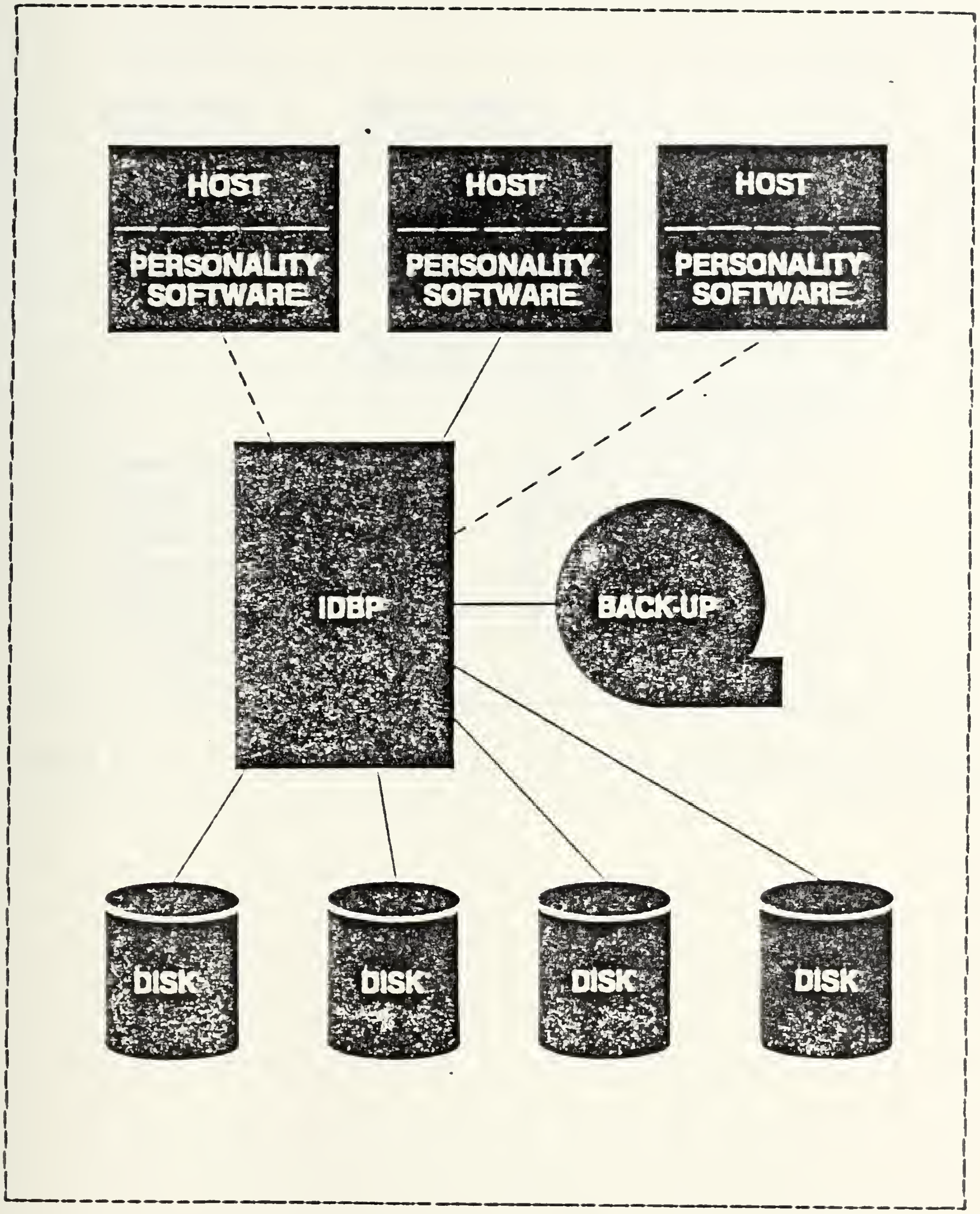
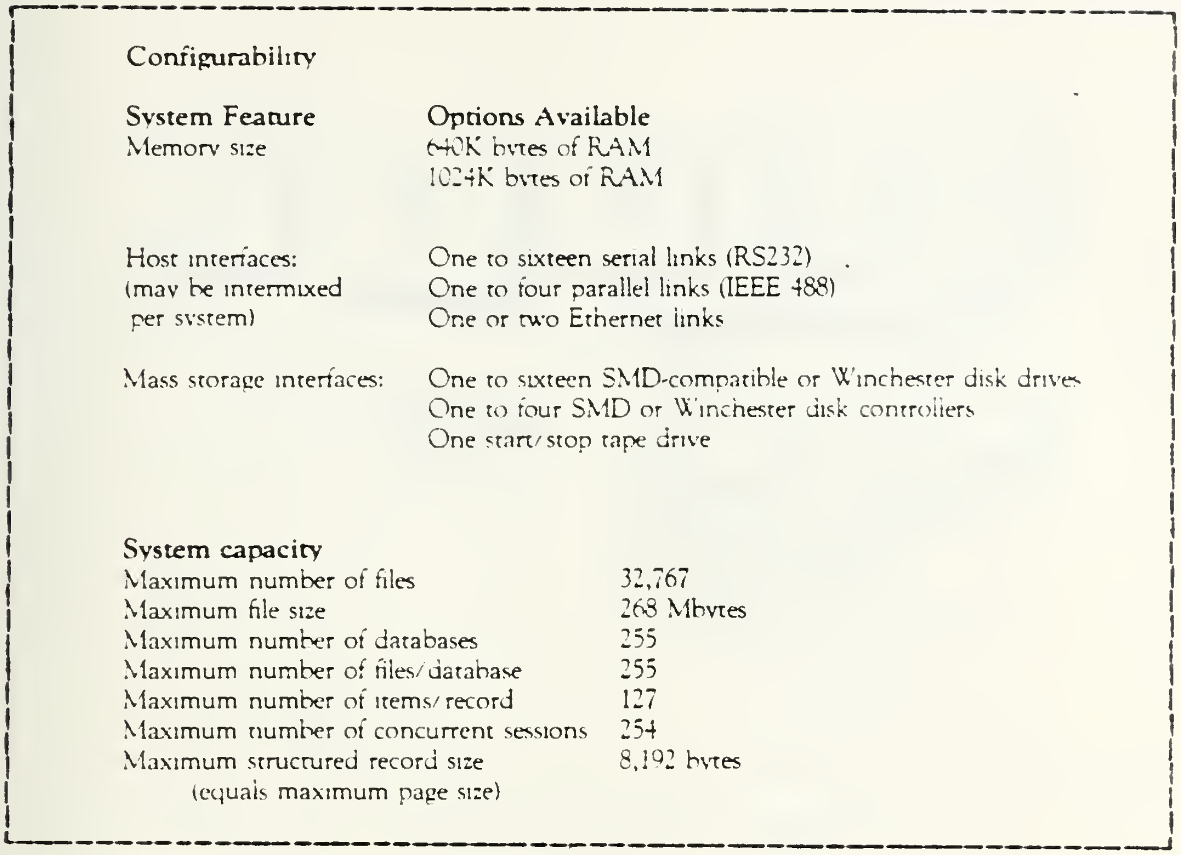


Figure 4.6 The iDBP System Architecture.



Configurability

System Feature	Options Available
Memory size	640K bytes of RAM 1024K bytes of RAM
Host interfaces: (may be intermixed per system)	One to sixteen serial links (RS232) One to four parallel links (IEEE 488) One or two Ethernet links
Mass storage interfaces:	One to sixteen SMD-compatible or Winchester disk drives One to four SMD or Winchester disk controllers One start/stop tape drive

System capacity

Maximum number of files	32,767
Maximum file size	268 Mbytes
Maximum number of databases	255
Maximum number of files/database	255
Maximum number of items/record	127
Maximum number of concurrent sessions	254
Maximum structured record size (equals maximum page size)	8,192 bytes

Figure 4.7 Configurability of the iDBP and it's System Capacities.

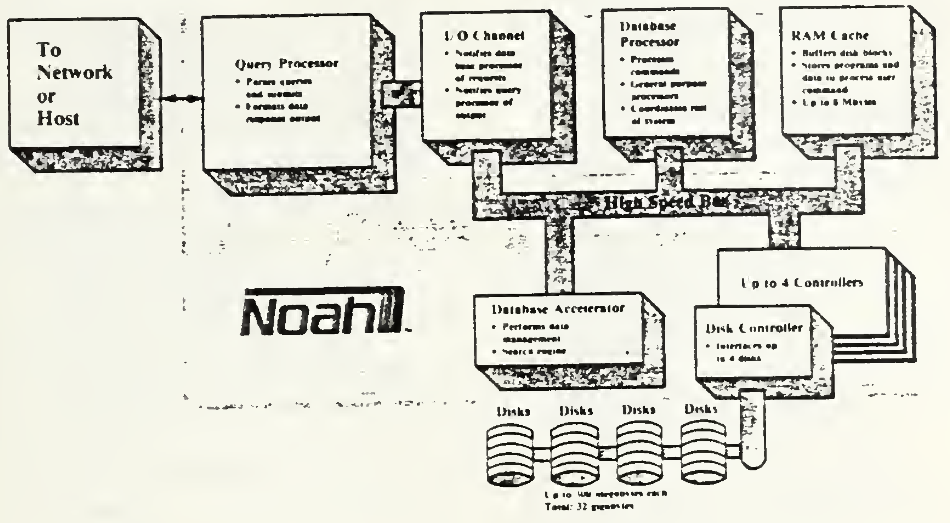
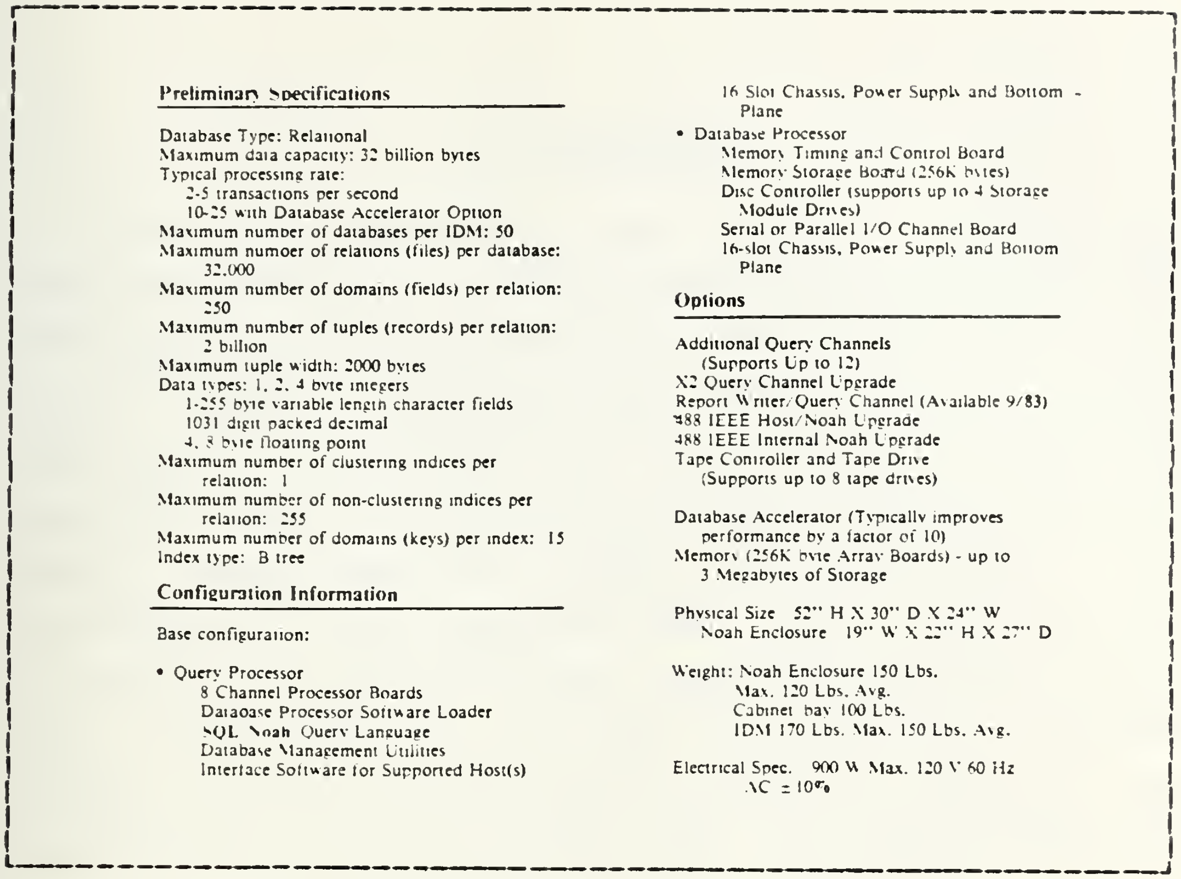

To Network or Host
Query Processor
• Parses queries and requests
• Formats data response output
I/O Channel
• Notifies data base processor of requests
• Notifies query processor of output
Database Processor
• Processes commands
• General purpose processors
• Coordinates rest of system
RAM Cache
• Buffers disk blocks
• Stores programs and data to process user command
• Up to 8 Mbytes
High Speed Bus
Noah
Up to 4 Controllers
Database Accelerator
• Performs data management
• Search engine
Disk Controller
• Interfaces up to 4 disks
Disks
Disks
Disks
Disks
Up to 500 megabytes each
Total: 32 gigabytes

Figure 4.8 NOAH's Hardware Configuration.



Preliminary Specifications

Database Type: Relational
Maximum data capacity: 32 billion bytes
Typical processing rate:
2-5 transactions per second
10-25 with Database Accelerator Option
Maximum number of databases per IDM: 50
Maximum number of relations (files) per database: 32,000
Maximum number of domains (fields) per relation: 250
Maximum number of tuples (records) per relation: 2 billion
Maximum tuple width: 2000 bytes
Data types: 1, 2, 4 byte integers
1-255 byte variable length character fields
1031 digit packed decimal
4, 8 byte floating point
Maximum number of clustering indices per relation: 1
Maximum number of non-clustering indices per relation: 255
Maximum number of domains (keys) per index: 15
Index type: B tree

Configuration Information

Base configuration:

- Query Processor
  - 8 Channel Processor Boards
  - Database Processor Software Loader
  - SQL Noah Query Language
  - Database Management Utilities
  - Interface Software for Supported Host(s)
  - 16 Slot Chassis, Power Supply and Bottom Plane
- Database Processor
  - Memory Timing and Control Board
  - Memory Storage Board (256K bytes)
  - Disc Controller (supports up to 4 Storage Module Drives)
  - Serial or Parallel I/O Channel Board
  - 16-slot Chassis, Power Supply and Bottom Plane

Options

Additional Query Channels
(Supports Up to 12)
X2 Query Channel Upgrade
Report Writer/Query Channel (Available 9/83)
488 IEEE Host/Noah Upgrade
488 IEEE Internal Noah Upgrade
Tape Controller and Tape Drive
(Supports up to 8 tape drives)

Database Accelerator (Typically improves performance by a factor of 10)
Memory (256K byte Array Boards) - up to 3 Megabytes of Storage

Physical Size 52" H X 30" D X 24" W
Noah Enclosure 19" W X 22" H X 27" D

Weight: Noah Enclosure 150 Lbs.
Max. 120 Lbs. Avg.
Cabinet bay 100 Lbs.
IDM 170 Lbs. Max. 150 Lbs. Avg.

Electrical Spec. 900 W Max. 120 V 60 Hz AC ± 10%

Figure 4.9 Specifications and Configuration Information.

V. CONCLUSIONS

Distributed processing and computer networks are enabling computers to use programs and data stored in computers at different locations. The SPLICE project is one of those projects in which these advances will be incorporated. Growth in communications, minicomputers and microcomputers is making the use of distributed processing possible. Many of the jobs which used to be done on large, heavily shared computers can now be done on stand alone minicomputers or microcomputers [Ref. 15]. The advantages of distributed information has received increasing attention. The recognition of these advantages has provided impetus to work on distributed systems. However, the complexities of such systems must be investigated and resolved if these systems are to work effectively.

This thesis only covered a small portion of SPLICE, the Database Management Module. A conceptual design of the database for SPLICE was given. Since the database will be one which will contain data on Supply parts, the attributes given in each relation were carefully chosen to reflect information needed in an inventory system. The relations which were designed are long. Shorter relations can be joined to produce the same attributes. Joins, however, can be time consuming. Therefore, the longer relations, which do not take as much time to provide the attributes needed to answer a query, were utilized.

Among the problems of a distributed system are access control and security. The use of encryption devices, user accounts and passwords are the minimum in security features which must be incorporated into SPLICE.

Concurrent updates in a distributed environment can cause a problem. However, deadlocks and livelocks can possibly be handled by using locking strategies. Besides the concurrent update problem, recovery from crashes must be considered. When a system fails, the number of transactions which were completed is unknown. There must be some type of logs or journals which will help to determine not only which transactions were completed, but also the source of the failure. All of this has to be considered in light of the fact that other systems in a distributed network must continue to function reguardless of the fact that one computer has failed.

Locking data is another problem within a local area network. Whether data is local or global, centralized or partitioned, will determine the magnitude of the problem. Accessing data as well as updating data can be a big problem especially if locks must be placed on the data.

Finally, alternative hardware considerations for SPLICE were discussed. The database computers were proposed as an alternative to conventional computers. The different approaches to database computers were given (back-end processor, intelligent peripheral, storage hierarchy, network node) along with a brief description of each. Also, several models of database computers were presented with their functions and architectural configurations. From the information presented on these database computers, all of which were relational, we found that they have features which could be very useful for SPLICE. The fact that they are able to directly access the content of a data item as well as being able to offload database management systems, make them very attractive alternatives. Some of the database computers are even able to interface with existing data-bases. We must, however, take into account the disadvantages of database computers. All thing considered, database computers seem to be a very viable alternative for SPLICE.

LIST OF REFERENCES

1. Schneidewind, Norman F., Methods for Interconnection Local Networks to Long Distance Networks, Code 54Ss, Naval Postgraduate School, Monterey, CA, February 1983

2. Tanenbaum, Andrew S., Computer Networks, Vrije Universiteit, Amsterdam, the Netherland, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1981

3. Schneidewind, Norman F., Functional Design of a Local Area Network for the Stock Point Logistics Integrated Communications Environment, NPS-54-82-003, Naval Postgraduate School, Monterey, CA, December, 1982

4. Navy Fleet Material Support Office, Executive Handbook, UADPS for Stock Points, Mechanicsburg, PA, January 30, 1980

5. Reinhart, Joseph N. III and Arana, Ricardo, Database and Terminal Management Functional Design Specifications in Support of Stock Point Logistics Integrated Communication Environment (SPLICE), Naval Postgraduate School, Monterey, CA, June 1982

6. Ullman, Jeffrey D., Principles of Database Systems, Second Edition, Computer Science Press, Inc., Stanford University, 1982

7. Date, C. J., An introduction to Database Systems, Third Edition, Addison-Wesley Publishing Company, 1981

8. Kroenke, David M., Busines Computers, An Introduction, Mitchell Publishing, Inc., Santa Cruz, CA, 1981

9. Champine, George A., "Four Approaches to a Data Base Computer" Datamation, December 1978

10. Banerjee, Jayanta, Hsiao, David K. (Senior member, IEEE) and Kannan, Krishnamurthi, "DBC - A Database Computer for Very Large Databases", IEEE Transactions of Computers, Vol. c-28, No.6, June 1979

11. Matabarba, Frank J.,"Data Base Machines-It's About Time!", Department of the Navy, Naval Data Automation Command, Washington, D.C.

12. Britton Lee, Inc., IDM System 300/600 Relational Database Management Systems, Los Gatos, CA, May 1982

13. Intel Corporation, Database Processor iDBP 86/440, 1982

14. Bridges, Terry, CDP,"Data Base Machines - What and Why", Database Management, November 1982

15. Martin, James, Computer Networks and Distributed Processing: Software, Computer Networks and Distributed Processing:Software,Techniques, and Architecture, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1981

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Prof. Norman F. Schneidewind Code 54Ss Administrative Sciences Department Naval Postgraduate School Monterey, CA 93940	1
2.	Prof. David Hsaio Code 52 Computer Science Department Naval Postgraduate School Monterey, CA 93940	1
3.	Computer Center Library Code 0141 Naval Postgraduate School Monterey, CA 93949	2
4.	Computer Science Department Code 52 Naval Postgraduate School Monterey, CA 93940	1
4.	LCDR Robert Jackson III Supply Officer Code 42 Naval Postgraduate School Monterey, CA 93940	1
5.	LT Gracie Thompson 1038 First Street No. 7 Monterey, CA 93940	1
6.	LT E. Jean Dixon 417 Secession Ave. Abbeville, S.C. 29620	3
7.	Defense Technical Information Center Cameron Station Alexandria, VA 23314	2
8.	Knox Library Code 0142 Naval Postgraduate School Monterey, CA 93940	2
9.	Prof. Dushan Badal Code 52Zd Computer Science Department Naval Postgraduate School Monterey, CA 93940	1
10.	LCDR Ted Case Fleet Material Support Office Code 94L Mechanicsburg, PA 17055	1

11.	Prof. Dan Dolk Code 54Dk Administrative Sciences Department Naval Postgraduate School Monterey, CA 93940	1
12.	LCDR Dana Fuller Commander, Naval Supply Systems Command Code 0415A Washington, D. C. 20376	1
13.	Ms. Mary Willoughby P. O. Box 94 Mendocino, CA 95460	1